NTT DATA Group Corporation

Cryptographic Library for OpenSSL powered by NTT DATA

FIPS 140-3 Non-Proprietary Security Policy

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document explains the non-proprietary FIPS 140-3 Security Policy for the "Cryptographic Library for OpenSSL powered by NTT DATA" (hereinafter referred to as "the cryptographic module") version 1.0.

This document describes the security rules that derive from the requirements of the FIPS 140-3 standard and specifies the security rules under which the cryptographic module operates. This non-proprietary security policy may only be reproduced and distributed perfectly in its entirety, including this notice. All other documents are the property of their respective authors.

1.2 Security Levels

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	N/A
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	1
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The cryptographic module is a multi-chip standalone cryptographic software library that provides cryptographic services executed in a cryptographically protected region in Intel SGX memory to applications operating outside the Intel SGX environment through a C language application programming interface (API).

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

The cryptographic module is based on the OSS cryptographic products "Crypto API Toolkit" and "OpenSSL," which provide PKCS#11 functionality for Intel SGX.
The cryptographic boundary of the cryptographic module consists of a part corresponding to the OpenSSL FIPS Provider, which is implemented using the provider architecture adopted from OpenSSL version 3, and a module that verifies the operation of cryptographic processing within the FIPS140-3 certification scope implemented in said part corresponding to OpenSSL FIPS Provider. The combination of these two components constitutes the FIPS 140-3 certification scope (crypto boundary) of the cryptographic module.
The "Crypto API Toolkit" incorporates the part corresponding to OpenSSL FIPS Provider and the module that verifies the operation of cryptographic processing within the scope of FIPS140-3 certification, which together constitute this cryptographic boundary.

Tested Operational Environment's Physical Perimeter (TOEPP):

Figure 1 shows the cryptographic boundary and operating environment of the cryptographic module, as well as the flow of information between the cryptographic module and the application (shown by arrows).

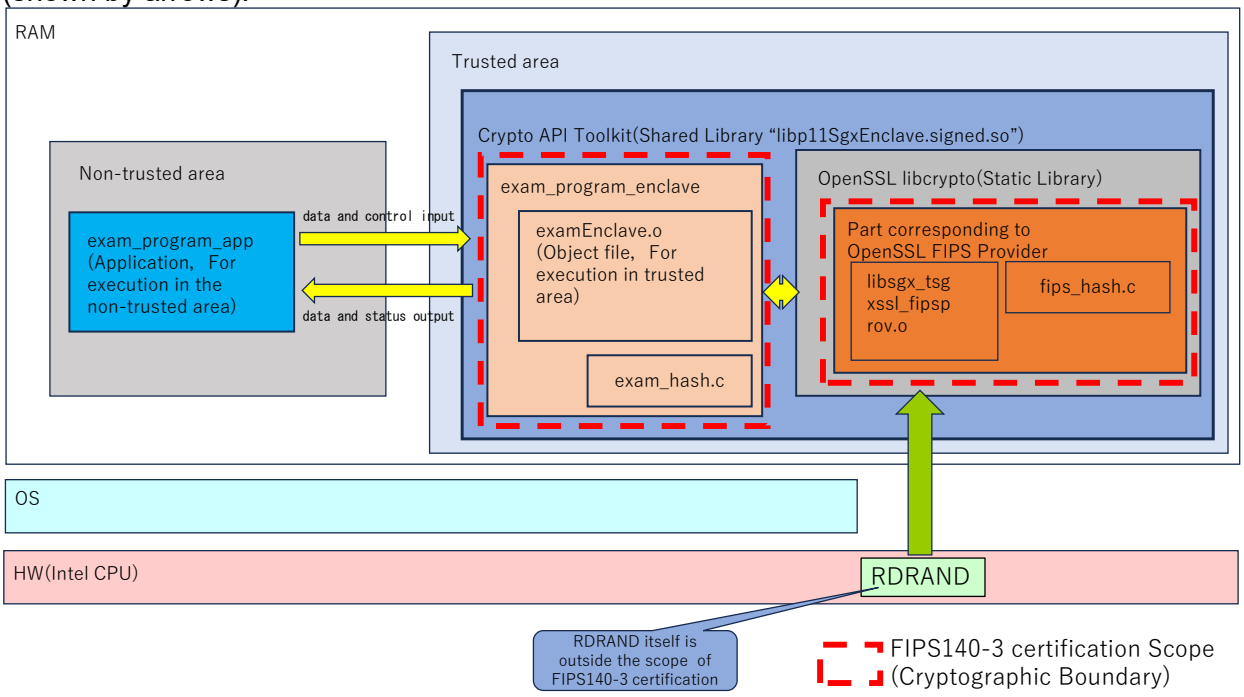

Figure 1: Block Diagram

"Part corresponding to OpenSSL FIPS Provider" in Figure 1 consists of the object file "libsgx_tsgxssl_fipsprov.o" that implements cryptographic functionality within the FIPS 140-3 certification scope and the source file "fips_hash.c" that stores the HMAC values used for integrity tests.

"exam_program_enclave" section in Figure 1 is composed of the object file "examEnclave.o," which is provided with functionality to verify cryptographic operations, and the source file "exam_hash.c," which stores the HMAC values used for integrity tests.

“Part corresponding to OpenSSL FIPS Provider” and "exam_program_enclave" will eventually be incorporated into the Crypto API Toolkit's shared library ”libp11SgxEnclave.signed.so”.

In addition, Figure 1 also shows that the cryptographic module receives a LOAD command containing entropy obtained from an entropy source (an Intel CPU processor with the RDRAND instruction). However, since the RDRAND instruction itself is outside the scope of development, it is excluded from the FIPS 140-3 certification scope. For more information about the entropy generated by the RDRAND instruction, see "2.8 RBG and Entropy."

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

N/A for this module.

Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets):

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libp11SgxEnclave.signed.so	1.0	N/A	HMAC-SHA2-256

Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
MIRACLE Linux 8.8 (64bit)	FR2100TX model 700	Xeon W-1270TE (2.0GHz)	No	N/A	1.0

Table 3: Tested Operational Environments - Software, Firmware, Hybrid

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:

N/A for this module.

2.3 Excluded Components

No components are exempt from the requirements of the FIPS 140-3 standard.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode	Automatically input whenever an approved service is requested	Approved	Equivalent to the indicator of the requested service

Table 4: Modes List and Description

If the cryptographic algorithms of the integrity tests and CASTs performed when the cryptographic module is loaded are successful, the cryptographic module automatically enters Approved mode, and cryptographic services using cryptographic algorithms become available via the logical interface. Conversely, when a cryptographic module is unloaded, all cryptographic services are no longer available.

Mode Change Instructions and Status:

Cryptographic services are only available in Approved mode. The status indicators for the mode of operation are the same as those for the requested service.

Degraded Mode Description:

This cryptographic module does not implement Degraded mode.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A6710	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A6710	Key Length - 128, 192, 256	SP 800-38C
AES-CFB1	A6710	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A6710	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A6710	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-CMAC	A6710	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A6710	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A6710	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A6710	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 192, 256	SP 800-38D
AES-OFB	A6710	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A6710	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A6710	Prediction Resistance - No, Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-5)	A6710	Curve - P-256, P-384, P-521 Secret Generation Mode - extra bits	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A6710	Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512, SHA2-512/256 Component - No	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A6710	Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512, SHA2-512/256	FIPS 186-5
EDDSA KeyGen	A6710	Curve - ED-25519, ED-448	FIPS 186-5
EDDSA SigGen	A6710	Curve - ED-25519, ED-448 PreHash - No Pure - Yes	FIPS 186-5
EDDSA SigVer	A6710	Curve - ED-25519, ED-448 PreHash - No Pure - Yes	FIPS 186-5
Hash DRBG	A6710	Prediction Resistance - No, Yes Mode - SHA2-256, SHA2-384, SHA2-512, SHA2-512/256, SHA3-256, SHA3-384, SHA3-512	SP 800-90A Rev. 1
HMAC DRBG	A6710	Prediction Resistance - No, Yes Mode - SHA2-256, SHA2-384, SHA2-512, SHA2-512/256, SHA3-256, SHA3-384, SHA3-512	SP 800-90A Rev. 1
HMAC-SHA2-256	A6710	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A6710	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A6710	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
HMAC-SHA2-512/256	A6710	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-256	A6710	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-384	A6710	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-512	A6710	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
KAS-ECC CDH-Component SP800-56Ar3 (CVL)	A6710	Function - Full Public Key Validation, Key Pair Generation, Partial Public Key Validation Curve - P-256, P-384, P-521	SP 800-56A Rev. 3
KAS-ECC-SSC Sp800-56Ar3	A6710	Domain Parameter Generation Methods - P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KTS-IFC	A6710	Modulo - 2048, 3072, 4096 Key Generation Methods - rsakpg1-crt Scheme - KTS-OAEP-basic - KAS Role - initiator, responder Key Transport Method - Key Length - 1024	SP 800-56B Rev. 2
RSA KeyGen (FIPS186-5)	A6710	Key Generation Mode - probableWithProbableAux Modulo - 2048, 3072, 4096 Primality Tests - 2powSecStr Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A6710	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-5)	A6710	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
SHA2-256	A6710	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A6710	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A6710	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512/256	A6710	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA3-256	A6710	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A6710	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A6710	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHAKE-128	A6710	Output Length - Output Length: 16-65536	FIPS 202

Algorithm	CAVP Cert	Properties	Reference
SHAKE-256	A6710	Output Length - Output Length: 16-65536	FIPS 202

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

N/A for this module.

The cryptographic module does not implement vendor-affirmed cryptographic algorithms.

Non-Approved, Allowed Algorithms:

N/A for this module.

The cryptographic module does not implement non-approved cryptographic algorithms.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

The module does not implement any non-approved algorithms allowed in the Approved mode of operation with no security claimed.

Non-Approved, Not Allowed Algorithms:

N/A for this module.

The cryptographic module does not implement non-approved cryptographic algorithms without security claims.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Symmetric Encryption with AES	BC-UnAuth	Symmetric encryption using AES	AES-CBC:128:128, 192, 256 bits AES-CFB1:128, 192, 256 bits AES-CFB128:128, 192, 256 bits AES-CFB8:128, 192, 256 bits	AES-CBC: (A6710) AES-CFB1: (A6710) AES-CFB128: (A6710) AES-CFB8: (A6710) AES-CTR: (A6710)

Name	Type	Description	Properties	Algorithms
			AES-CTR:128, 192, 256 bits AES-OFB:128, 192, 256 bits AES-XTS Testing Revision 2.0:128, 256 bits	AES-OFB: (A6710) AES-XTS Testing Revision 2.0: (A6710)
Symmetric Decryption with AES	BC-UnAuth	Symmetric decryption using AES	AES-CBC:128, 192, 256 bits AES-CFB1:128, 192, 256 bits AES-CFB128:128, 192, 256 bits AES-CFB8:128, 192, 256 bits AES-CTR:128, 192, 256 bits AES-OFB:128, 192, 256 bits AES-XTS Testing Revision 2.0:128, 256 bits	AES-CBC: (A6710) AES-CFB1: (A6710) AES-CFB128: (A6710) AES-CFB8: (A6710) AES-CTR: (A6710) AES-OFB: (A6710) AES-XTS Testing Revision 2.0: (A6710)
Random Number Generation	DRBG	Random Number Generation (IG D.R compliant)	Counter DRBG:128, 192, 256 bits HMAC DRBG:128, 256 bits Hash DRBG:128, 256 bits	Counter DRBG: (A6710) HMAC DRBG: (A6710) Hash DRBG: (A6710)
Signature Generation	DigSig-SigGen	Signature Generation	ECDSA SigGen (FIPS186-5):P-256, P-384, P521 (128, 192, 256 bits) EDDSA SigGen :(FIPS186-5): ED448, ED25519 (128, 256 bits) RSA SigGen (FIPS186-5):2048, 3072, 4096 bits (112, 128, 140 bits)	ECDSA SigGen (FIPS186-5): (A6710) EDDSA SigGen: (A6710) RSA SigGen (FIPS186-5): (A6710)
Signature Verification	DigSig-SigVer	Signature Verification	ECDSA SigVer (FIPS186-5):P-256, P-384, P521 (128, 192, 256	ECDSA SigVer (FIPS186-5): (A6710) EDDSA SigVer:

Name	Type	Description	Properties	Algorithms
			bits) EDDSA SigVer (FIPS186-5):ED448, ED25519 (128, 256 bits) RSA SigVer (FIPS186-5):2048, 3072, 4096 bits (112, 128, 140 bits)	(A6710) RSA SigVer (FIPS186-5): (A6710)
Message Authentication Code	MAC	Message Authentication Code	HMAC hashes:SHA2-256, SHA2-384, SHA2-512, SHA2-512/256, SHA3-256, SHA3-384, SHA3-512 AES key:128, 192, 256 bits	HMAC-SHA2-256: (A6710) HMAC-SHA2-384: (A6710) HMAC-SHA2-512: (A6710) HMAC-SHA2-512/256: (A6710) HMAC-SHA3-256: (A6710) HMAC-SHA3-384: (A6710) HMAC-SHA3-512: (A6710) AES-CMAC: (A6710) AES-GMAC: (A6710) Counter DRBG: 256bit Counter DRBG: (A6710)
Message digest	SHA XOF	Message digest		SHA2-256: (A6710) SHA2-384: (A6710) SHA2-512: (A6710) SHA2-512/256: (A6710) SHA3-256: (A6710) SHA3-384: (A6710) SHA3-512: (A6710) SHAKE-128: (A6710)

Name	Type	Description	Properties	Algorithms
				SHAKE-256: (A6710)
Authenticated Symmetric Encryption	BC-Auth	Authenticated Symmetric Encryption	Key:128, 192, 256 bits	AES-CCM: (A6710) AES-GCM: (A6710) Counter DRBG: 256bit Counter DRBG: (A6710)
Authenticated Symmetric Decryption	BC-Auth	Authenticated Symmetric Decryption	Key:128, 192, 256 bits	AES-CCM: (A6710) AES-GCM: (A6710)
Key Pair Generation with ECDSA	AsymKeyPair-KeyGen	Key Pair Generation for ECDSA	ECDSA KeyGen (FIPS186-5):P-256, P-384, P521 (128, 192, 256 bits)	ECDSA KeyGen (FIPS186-5): (A6710) Counter DRBG: 256bit Counter DRBG: (A6710)
Key Pair Generation with EDDSA	AsymKeyPair-KeyGen	Key Pair Generation for EDDSA	EDDSA KeyGen (FIPS186-5):ED448, ED25519(128, 256 bits)	EDDSA KeyGen: (A6710) Counter DRBG: 256bit Counter DRBG: (A6710)
Key Pair Generation with RSA	AsymKeyPair-KeyGen	Key Pair Generation for RSA	RSA KeyGen (FIPS186-5):2048, 3072, 4096 bits (112, 128, 140 bits)	RSA KeyGen (FIPS186-5): (A6710) Counter DRBG: 256bit Counter DRBG: (A6710)
KAS-SSC Shared Secret Computation with ECDH	KAS-SSC	Shared Secret Computation using EC Diffie-Hellman	IG:IG D.F Scenario 2, path (1)	KAS-ECC-SSC Sp800-56Ar3: (A6710) KAS-ECC CDH-Component SP800-56Ar3: (A6710)
Key transport	AsymKeyPair-Decap AsymKeyPair-Encap	Key transport	KTS-IFC KTS-OAEP-basic SP 800-56B Rev.2:2048, 3072, 4096 bits (112, 128, 140 bits)	KTS-IFC: (A6710)

Name	Type	Description	Properties	Algorithms
			Key Generation Methods:rsakpg1-crt KAS Role:initiator, responder	

Table 6: Security Function Implementations

2.7 Algorithm Specific Information

SHA-3:

The cryptographic module implements the SHA-3 algorithm both as a stand-alone and as part of a high-level algorithm (compliant with FIPS 140-3 IG C.C). As explained in "2.6 Security Function Implementations," cryptographic algorithms that use SHA-3 include RSA signature generation and verification, ECDSA signature generation and verification, and HMAC. Furthermore, implementation of the expandible output functions SHAKE128 and SHAKE256 has been verified to work standalone.

RSA sigGen, sigVer:

The module implements the approved RSA modulus sizes of 2048, 3072, and 4096 bits for signature generation. For signature verification, the module implements the approved RSA modulus size of 2048, 3072, and 4096 bits. These RSA modulus sizes comply with FIPS 140-3 IG C.F and have been tested by CAVP.

EdDSA:

The cryptographic module complies with FIPS 140-3 IG C.K as follows.

- Key pair generation, signature generation, and signature verification have been tested and verified with the elliptic curves ED448 and ED25519 that can be used by CAVP tests.
- ED448 signature generation and signature verification generates SHAKE256 hash value of the message to be signed, and ED25519 signature generation and signature verification generates SHA2-512 hash value of the message to be signed.

Key Agreement:

The cryptographic module implements the following approved key agreement methods, which have been tested and verified by CAVP.

- KAS-ECC-SSC per SP 800-56A Rev. 3 (FIPS 140-3 IG D.F Scenario 2, path 1)
- KAS ECC CDH-Component that is only used internally within the context of a SP 800-56Arev3 KAS as required in FIPS 140-3 IG 2.4.B.

The module has obtained the key agreement assurance required by FIPS 140-3 IG D.F.

- SP 800-56A Rev. 3 in accordance with Section 5.6.2 (Section 5.6.2.1.2, 5.6.2.1.4, 5.6.2.3.3, 5.6.2.3.4 are applicable).

The module does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

Key Transport:

The cryptographic module implements the following key distribution method, which has been tested and verified by CAVP.

- KTS-OAEP-basic with modulus sizes 2048, 3072, and 4096 bits, with rsakpg1-crt as the RSA key generation method (key confirmation is not supported), with both the encapsulation and un-encapsulation methods supported, and with assurances per SP 800-56B Rev.2 Section 6.4.1.2.3. (FIPS 140-3 IG D.G)
- The module does not establish SSPs using an approved key transport scheme (KTS).
  However, it does offer approved authenticated algorithms that can be used by an external operator/application as part of an approved KTS.

AES-GCM:

The cryptographic module provides AES GCM that accepts external IV generated by the operator using approved DRBG. The IV generation with approved DRBG must be provided as an API of the cryptographic module, but the length of the IV must always be 96 bits or higher to comply with Scenario 2 of FIPS 140-3 IG C.H.

AES XTS:

According to SP 800-38E, the AES-XTS algorithm is used to ensure confidentiality on storage devices. The module complies with FIPS 140-3 IG C.I in the following ways:

- By following instruction in Section 11.2, Key_1 and Key_2 are separately generated according to the rules for component symmetric keys in SP 800-133 Rev.2, Section 6.3.
- The module explicitly checks that Key_1 $\neq$ Key_2 before processing any data with the key in the AES-XTS algorithm, and if Key_1 = Key_2, the module returns an error code.

2.8 RBG and Entropy

N/A for this module.

N/A for this module.

In the cryptographic module, entropy and seed material for the SP 800-90Ar1 DRBG are provided by an application that executes an externally called RDRAND instruction (not the cryptographic module) and is outside the cryptographic boundary of the cryptographic module. The cryptographic module receives a LOAD command containing entropy obtained from an entropy source (an Intel CPU processor with the RDRAND instruction).

Note that RDRAND is an instruction that returns a random number from the Intel on-chip hardware random number generator, which is seeded by an on-chip entropy source. The generation of SSPs that require random numbers uses random numbers from a DRBG that complies with SP 800-90 Ar1. The entropy source that is input to the random number generation of that DRBG must provide at least 112 bits of entropy in order to satisfy the security strength required for the random number generation mechanism specified in SP 800-90Ar1. Therefore, the minimum number of bits of entropy requested when the module makes a call to the DRBG is at least 112 bits.

Per the IG 9.3.A Entropy Caveats, the following caveat applies: No assurance of the minimum strength of generated SSPs (e.g., keys).

The specifications of the DRBG implemented in the cryptographic module are shown below.

Counter DRBG:

Entropy Size：128-384bit

Seed：256, 320, 348 bits

DRBG internal state(V value, Key)：128, 192, 256 bits

Used By：Random Number Generation

AES-GMAC
AES-GCM
ECDSA KeyGen
ECDSA SigGen
EDDSA KeyGen
RSA KeyGen

Hash DRBG:

Entropy Size：128-384bit

Seed：440, 888 bits

DRBG internal state (V value, C value)：440, 888 bits

Used By：Random Number Generation

HMAC DRBG:

Entropy Size：128-384bit

Seed：160, 256, 512 bits

DRBG internal state(V value, Key)：128, 192, 256 bits

Used By：Random Number Generation

2.9 Key Generation

The cryptographic module implements cryptographic key generation that complies with SP 800-133r2. If random values are required, they are obtained from an SP 800-90Ar1-approved DRBG(Counter DRBG 256bit) that complies with Section 4 of SP 800-133r2.

- RSA key pair generation: Compliant with SP 800-133r2, Section 5.1, which maps to FIPS 186-5.
- ECC (ECDH and ECDSA) key pair generation: Compliant with SP 800-133r2, Section 5.1, which maps to FIPS 186-5.

In addition, the cryptographic module implements the following key pair generation methods:

- EDDSA key pair generation: Compliant with RFC8032, Section 5.1.5 (ED25519) and 5.2.5 (ED448), which are mapped to FIPS 186-5.

The generated key pair is output through the application program interface (API) coded in C on the cryptographic module, but intermediate values are explicitly set to zero after the service processing, without being output.

2.10 Key Establishment

The cryptographic module provides Elliptic-Curve Diffie-Hellman (ECDH) shared key generation that complies with SP 800-56Ar3, as per FIPS 140-3 IG D.F. Scenario 2 (1).
For ECDH, three NIST-defined elliptic curves (P-256, P-384, and P-521) are supported, providing 128-bit, 192-bit, and 256-bit security strengths as approved modes of operation.
To generate an ECDH shared key, it is necessary to generate an ECC key pair as described in "2.9 Key Generation." The ECC key pair generation uses random values obtained from the 256-bit Counter DRBG approved by SP 800-90Ar1.

2.11 Industry Protocols

The cryptographic module complies with FIPS 140-3 IG D.C references regarding support for industry protocols, but does not provide API entry points for schemes compliant with SP 800-56A Rev. 3 or for use with TLS, nor does it include complete implementation of these industry protocols. Therefore, it has not been tested against industry protocols by CAVP and CMVP.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API input parameters for data
N/A	Control Input	API function calls
N/A	Data Output	API output parameters for data
N/A	Status Output	Status Output API

Table 7: Ports and Interfaces

For the cryptographic module, control of the physical ports is outside the scope of the module. However, when the cryptographic module is performing self-tests or is in an error state, all output on the logical data output interface is prohibited, and only error return values are returned.
The module does not support a “control output” interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

The cryptographic module does not support authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 8: Roles

This module supports only the Crypto Officer role. The Crypto Officer role is always implicitly assumed by the operator of the cryptographic module when performing services.
The cryptographic module does not support simultaneous operation by multiple operators.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric Encryption	Symmetric encryption of an entry plaintext	API returns 1(FIPS_OK)	Plaintext, AES key, IV	Ciphertext	Symmetric Encryption with AES	Crypto Officer - AES key: W,E
Symmetric Decryption	Symmetric decryption of an entry ciphertext	API returns 1(FIPS_OK)	CipherPlaintext, AES key, IV	Plaintext	Symmetric Decryption with AES	Crypto Officer - AES key: W,E
Authenticated Encryption	Symmetric encryption of an entry plaintext, including AEAD modes (CCM, GCM)	API returns 1(FIPS_OK)	Plaintext, AES key, IV	Ciphertext, Tag	Authenticated Symmetric Encryption	Crypto Officer - AES key: W,E
Authenticated Decryption	Symmetric decryption of an entry	API returns 1(FIPS_OK)	CipherPlaintext, AES key, IV, Tag	Plaintext	Authenticated Symmetric Decryption	Crypto Officer - AES key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	ciphertext, including AEAD modes (CCM, GCM)					
Message Authentication Code	Compute a MAC	API returns 1(FIPS_OK)	Message, AES key or HMAC key	MAC output value.	Message Authentication Code	Crypto Officer - AES key: W,E - HMAC Key: W,E
Message Digest	Generate a message digest	API returns 1(FIPS_OK)	Message	Message digest	Message digest	Crypto Officer
Shared Secret Computation	Compute a shared secret	API returns 1(FIPS_OK)	key; EC private key, EC public key	Shared secret	KAS-SSC Shared Secret Computation with ECDH	Crypto Officer - EC Private key: W,E - EC Public key: W,E - Shared Secret: G,R - SP800-56Ar3 domain parameters: E
Signature Generation	Generate a digital signature	API returns 1(FIPS_OK)	Message, private key	Signature	Signature Generation	Crypto Officer - RSA private key: W,E - EC Private key: W,E - ED448 or ED 25519 Private key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Signature Verification	Verify a digital signature	API returns 1(FIPS_OK)	Message, public key, signature	Pass/fail result of verification	Signature Verification	Crypto Officer - RSA public key: W,E - EC Public key: W,E - ED448 or ED 25519 Public key: W,E
Key Pair Generation with ECDSA	Generate an ECDSA key pair	API returns 1(FIPS_OK)	Curve	EC key pair	Key Pair Generation with ECDSA	Crypto Officer - EC Private key: G,R - EC Public key: G,R - Intermediate Key Generation Value: G,R,W,Z
Key Pair Generation with EDDSA	Generate an EDDSA key pair	API returns 1(FIPS_OK)	Curve	EDDSA key pair	Key Pair Generation with EDDSA	Crypto Officer - ED448 or ED 25519 Private key: G,R - ED448 or ED 25519 Public key: G,R - Intermediate Key Generation Value: G,R,W,Z
Key Pair Generation with RSA	Generate an RSA key pair	API returns	Modulus bits	RSA key pair	Key Pair Generation with RSA	Crypto Officer - RSA

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		1(FIPS_OK)				private key: G,R - RSA public key: G,R - Intermediate Key Generation Value: G,R,W,Z
RSA Key Pair Generation for CAVP	Generate a RSA key pair for CAVP	API returns 1(FIPS_OK)	Modulus bits; Public Exponent; Prime Num(p, q, p-1, p-2, q-1, q-2)	RSA key pair	Key Pair Generation with RSA	Crypto Officer - RSA private key: G,R - RSA public key: G,R - Intermediate Key Generation Value: G,R,W,Z
Key transport(encryption)	Asymmetric encryption of an entry plaintext	API returns 1(FIPS_OK)	plaintext, public key	ciphertext	Key transport	Crypto Officer - RSA public key: W,E
Key transport(decryption)	Asymmetric decryption of an entry ciphertext	API returns 1(FIPS_OK)	ciphertext, private key	plaintext	Key transport	Crypto Officer - RSA private key: W,E
Random Number Generation	Generate random bytes	API returns 1(FIPS_OK)	Output length	Random bytes	Random Number Generation	Crypto Officer - Entropy input: W,E - DRBG internal state (V value, C value): G,W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- DRBG internal state (V value, Key): G,W,E - DRBG seed: G,W,E
Random Number Generation for CAVP	Generate random bytes for CAVP	API returns 1(FIPS_OK)	Output length; entropy_input; nonce; personalization; prediction resistance flag; additional input	Random bytes	Random Number Generation	Crypto Officer - Entropy input: W,E - DRBG internal state (V value, C value): G,W,E - DRBG internal state (V value, Key): G,W,E - DRBG seed: G,W,E
Show status	Show the current status of the module	API returns 1(FIPS_OK)	N/A	Module status	None	Crypto Officer
Show module name and version	Show module name and the version of the module	API returns 1(FIPS_OK)	N/A	Name and version information	None	Crypto Officer
Self-test	Perform CASTs and integrity test	None	N/A	Pass/fail result of self-tests	None	Crypto Officer
Zeroization	Zeroize SSPs.	None	Any SSP	N/A	None	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- AES key: Z - HMAC Key: Z - RSA private key: Z - RSA public key: Z - EC Private key: Z - EC Public key: Z - ED448 or ED 25519 Private key: Z - ED448 or ED 25519 Public key: Z - Shared Secret: Z - Entropy input: Z - DRBG seed: Z - DRBG internal state (V value, C value): Z - DRBG internal state (V value, Key): Z - SP800-56Ar3 domain parameters: Z

Table 9: Approved Services

The cryptographic module provides services to operators who bear the available roles. All services are detailed in the API documents (manual pages). When specifying the type of access that a service has for each SSP, the following rules apply.

- Generate (G): The cryptographic module generates an SSP.
- Read (R): The SSP is read from the cryptographic module (Example: output to a higher-level application).
- Write (W): The SSP is updated, imported, or written to the cryptographic module.
- Execute (E): The cryptographic module uses the SSP when performing cryptographic operations.
- Zeroize (Z): The cryptographic module sets the SSP to zero.
- N/A: The cryptographic module does not access the SSP during operation.

To interact with the cryptographic module, the calling application must use the APIs provided by this cryptographic module (APIs beginning with "exam_"). Within these APIs, services are used via the OpenSSL EVP API layer. In addition, approved service indicators can be obtained using the API provided by the cryptographic module.
Furthermore, the cryptographic module provides an API to obtain the current status, name, and version.

4.4 Non-Approved Services

N/A for this module.

4.5 External Software/Firmware Loaded

N/A for this module.

5 Software/Firmware Security

5.1 Integrity Techniques

As a measure to ensure software security in the cryptographic module, the integrity of the modules belonging to the two cryptographic boundaries indicated in Figure 1 in 2.1 Description is checked (integrity test) when execution starts.
Specifically, first, the HMAC-SHA2-256 value calculated from "libsgx_tsgxssl_fipsprov.o", which corresponds to the part of OpenSSL FIPS Provider indicated in Figure 1, is compared with the HMAC-SHA2-256 value stored in "fips_hash.c" (the former is calculated at runtime, and the latter is calculated at build time).
Next, the HMAC-SHA2-256 value calculated from "examEnclave.o" in "exam_program_enclave" in Figure 1 is compared with the HMAC-SHA2-256 value stored in "exam_hash.c" (the former is calculated at runtime, and the latter is calculated at build time).

5.2 Initiate on Demand

The integrity check (integrity test) described in "5.1 Integrity Techniques" is executed as part of the pre-operation self-test performed within the cryptographic module load API (exam_start_fips) that is provided by the cryptographic module. If necessary, should you want to perform this integrity check again, execute the cryptographic module unload API (exam_end_fips) once, and then execute the cryptographic module load API (exam_start_fips) again.

5.3 Open-Source Parameters

To build the cryptographic module, the compiler gcc version 8.5 is necessary.
There are no specific control parameters required to build the cryptographic module.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

No operating environment restrictions are required for operation in Approved mode. All the conditions for the module to operate in Approved mode are stated in "2.4 Modes of Operation."

How Requirements are Satisfied:

All SSPs contained within the cryptographic module are protected by process isolation and memory isolation mechanisms, and the cryptographic module has sole control over these SSPs.

6.2 Configuration Settings and Restrictions

The cryptographic module must be installed as described in "11.1 Installation, Initialization, and Startup Procedures." When properly installed, an operating system provides process isolation and memory protection mechanisms that ensure that memory access is properly separated between processes on the system. Each process has control over its own data, and uncontrolled access to the data of other processes is prevented. Instrumentation tools like gdb and strace, frameworks like ftrace and systemtap provided by Linux, and other tracing mechanisms should not be used in a production environment (as the use of these tools results in the cryptographic module being run in a production environment that has not been validated).

7 Physical Security

N/A for this module.
Since the cryptographic module consists only of software, this section is not applicable.

8 Non-Invasive Security

N/A for this module.
The cryptographic module does not require non-invasive security techniques.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	Random access memory	Dynamic

Table 10: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API input parameters	Operator calling application	RAM	Plaintext	Manual	Electronic	
API output parameters	RAM	Operator calling application	Plaintext	Manual	Electronic	

Table 11: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Cleanse	Caller invocation of OPENSSL_cleanse.	Overwrites with zeros	Caller invocation of OPENSSL_cleanse
Teardown	Module unload - invokes cleanse internally.	Overwrites with zeros	Caller invocation of OPENSSL_cleanse

Table 12: SSP Zeroization Methods

Of the SSP/keys generated within the cryptographic module, those temporarily stored within the cryptographic module are zeroized (overwritten with 0) when they become unnecessary.
Specifically, the zeroization of the SSP is performed within the OpenSSL "Free cipher handle" family of functions that are called within the APIs that perform the services of the cryptographic module (APIs beginning with "exam_").
Furthermore, since zeroization is performed in the memory, the time required for zeroization is close to zero seconds, essentially preventing the possibility for the SSP to be compromised.

The SSP provided to the application must be zeroized on the application side. For specific instructions, see (B) of "11.2 Administrator Guidance."

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	AES key used for encryption, decryption, and computing MAC tags	AES-XTS: 256, 512 bits Other modes: 128, 192, 256 bits - AES-XTS: 256, 512 bits Other modes: 128, 192, 256 bits	Symmetric key - CSP			Symmetric Encryption with AES Symmetric Decryption with AES Authenticated Symmetric Encryption Authenticated Symmetric Decryption
HMAC Key	Compute a MAC	112 - 524288 bits - 128 - 256 bits	Symmetric key - CSP			Message Authentication Code
RSA private key	RSA private key	2048, 3072, 4096 bits - 112, 128, 140 bits	Private key - CSP	Key Pair Generation with RSA		Signature Generation Key Pair Generation with RSA Key transport
RSA public key	RSA public key	2048, 3072, 4096 bits - 112, 128, 140 bits	Public key - PSP	Key Pair Generation with RSA		Signature Verification Key Pair Generation with RSA Key transport
EC Private key	EC Private key	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Key Pair Generation with ECDSA KAS-SSC Shared Secret Computation with ECDH		Signature Generation KAS-SSC Shared Secret Computation with ECDH
EC Public key	EC Public key	P-256, P-384, P-	Public key - PSP	Key Pair Generation		Signature Verification

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		521 - 128, 192, 256 bits		with ECDSA KAS-SSC Shared Secret Computation with ECDH		KAS-SSC Shared Secret Computation with ECDH
ED448 or ED 25519 Private key	ED448 or ED25519 Private key	ED448, ED25519 - 128, 256 bits	Private key - CSP	Key Pair Generation with EDDSA		Signature Generation
ED448 or ED 25519 Public key	ED448 or ED25519 Public key	ED448, ED25519 - 128, 256 bits	Public key - PSP	Key Pair Generation with EDDSA		Signature Verification
Shared Secret	Shared secret generated by ECDH shared	P-256, P-384, P-521 - 128, 192, 256 bits	Shared Secret - CSP		KAS-SSC Shared Secret Computation with ECDH	
Entropy input	An entropy input is a input data stream generated from entropy collected from outside the cryptographic module's boundary, and the entropy input data stream is used to seed the DRBG (IG D.L compliant)	128 - 384 bits - 128 - 384 bits	Entropy Input - CSP			Random Number Generation
DRBG seed	DRBG seed derived from entropy	Counter DRBG: 256, 320, 348 bits	Seed - CSP	Random Number Generation		Random Number Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	input (IG D.L compliant)	HMAC DRBG: 160, 256, 512 bits Hash DRBG: 440, 888 bits - Counter DRBG: 128, 192, 256 bits HMAC DRBG: 128, 192, 256 bits Hash DRBG: 128, 256 bits				
DRBG internal state (V value, C value)	Internal state of the Hash DRBG (IG D.L compliant)	440, 888 bits - 128, 256 bits	Internal state - CSP	Random Number Generation		Random Number Generation
DRBG internal state (V value, Key)	Internal state of the Counter DRBG and HMAC DRBG (IG D.L compliant)	Counter DRBG: 128, 192, 256 bits HMAC DRBG: 256, 384, 512 bits - Counter DRBG: 128, 192, 256 bits HMAC DRBG:128, 192, 256	Internal state - CSP	Random Number Generation		Random Number Generation
Intermediate Key Generation Value	Intermediate key pair generation value generated	224-4096 bits - 112, 256bits	Intermediate - CSP	Key Pair Generation with ECDSA Key Pair Generation	Key Pair Generation with ECDSA Key Pair Generation	

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	during key generation			with EDDSA Key Pair Generation with RSA	with EDDSA Key Pair Generation with RSA	
SP800-56Ar3 domain parameters	SP800-56Ar3 domain parameters	P-256, P-384, P-521 - 128, 192, 256 bits	domain parameters - PSP			KAS-SSC Shared Secret Computation with ECDH

Table 13: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters	RAM:Plaintext	From service invocation until cipherhandle is freed	Cleanse	
HMAC Key	API input parameters	RAM:Plaintext	From service invocation until cipherhandle is freed	Cleanse	
RSA private key	API input parameters API output parameters	RAM:Plaintext	From service invocation until cipherhandle is freed	Cleanse	RSA public key:Paired With Intermediate Key Generation Value:Generated From
RSA public key	API input parameters API output parameters	RAM:Plaintext	From service invocation until cipherhandle is freed	Cleanse	RSA private key:Paired With Intermediate Key Generation Value:Generated From
EC Private key	API input parameters API output parameters	RAM:Plaintext	From service invocation until cipherhandle is freed	Cleanse	EC Public key:Paired With Intermediate Key Generation Value:Generated From SP800-56Ar3 domain parameters:Generated From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
EC Public key	API input parameters API output parameters	RAM:Plaintext	From service invocation until cipherhandle is freed	Cleanse	Shared Secret:Paired With Intermediate Key Generation Value:Generated From SP800-56Ar3 domain parameters:Generated From
ED448 or ED 25519 Private key	API input parameters API output parameters	RAM:Plaintext	From service invocation until cipherhandle is freed	Cleanse	ED448 or ED 25519 Public key:Paired With Intermediate Key Generation Value:Generated From
ED448 or ED 25519 Public key	API input parameters API output parameters	RAM:Plaintext	From service invocation until cipherhandle is freed	Cleanse	ED448 or ED 25519 Private key:Paired With Intermediate Key Generation Value:Generated From
Shared Secret	API output parameters	RAM:Plaintext	From service invocation until cipherhandle is freed	Cleanse	EC Private key:Established By EC Public key:Established By
Entropy input		RAM:Plaintext	From generation until DRBG seed is created	Teardown	DRBG seed:Derived From
DRBG seed		RAM:Plaintext	While the DRBG is instantiated	Teardown	Entropy input:Derived From DRBG internal state (V value, C value):Generates DRBG internal state (V value, Key):Generates
DRBG internal state (V value, C value)		RAM:Plaintext	From DRBG instantiation until DRBG termination	Cleanse	DRBG seed:Generated From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
DRBG internal state (V value, Key)		RAM:Plaintext	From DRBG instantiation until DRBG termination	Cleanse	DRBG seed:Generated From
Intermediate Key Generation Value		RAM:Plaintext	From service invocation until API return	Cleanse	RSA private key:Generates RSA public key:Generates EC Private key:Generates EC Public key:Generates ED448 or ED 25519 Private key:Generates ED448 or ED 25519 Public key:Generates
SP800-56Ar3 domain parameters		RAM:Plaintext	From service invocation until cipherhandle is freed	Cleanse	EC Private key:Generates EC Public key:Generates

Table 14: SSP Table 2

9.6 Additional Information

The generation of SSPs that require random numbers uses random numbers from a DRBG that complies with SP 800-90. The entropy source that is input to the random number generation of that DRBG must provide at least 112 bits of entropy in order to satisfy the security strength required for the random number generation mechanism specified in SP 800-90Ar1. RDRAND is an instruction that returns a random number from the Intel on-chip hardware random number generator, which is seeded by an on-chip entropy source.

In the cryptographic module, entropy and seed material for the SP 800-90Ar1 DRBG are provided by an application that executes an externally called RDRAND instruction (not the cryptographic module) and is outside the cryptographic boundary of the cryptographic module. The cryptographic module receives a LOAD command containing entropy obtained from an entropy source (an Intel CPU processor with the RDRAND instruction).

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A6710)	256-bits key	Message authentication	SW/FW Integrity	Pre-Operational Self-Tests result(1 is success, 0 is error)	This test is performed on the two cryptographic boundaries shown in Figure 1, "Part corresponding to OpenSSL FIPS Provider" and "exam_program_enclave".

Table 15: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 (A6710)	HMAC-SHA2-256	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	key Message authentication	Before performing the SW integrity test, it is carried out on module load and SW/FW integrity.
SHA2-512 (A6710)	SHA2-512	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Message digest	Performed on module load.
SHA3-256 (A6710)	SHA3-256	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Message digest	Performed on module load.
AES-GCM-ENC	128bit	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Encrypt	Performed on module load.
AES-GCM-DEC	128bit	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Decrypt	Performed on module load.
AES-CBC (A6710)	128bit	KAT	CAST	Conditional Self-Tests result (1 is	Decrypt	Performed on module load.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				success, 0 is error)		
RSA SigGen (FIPS186-5) (A6710)	2048-bit with SHA2-256	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Sign	Performed on module load.
RSA SigVer (FIPS186-5) (A6710)	2048-bit with SHA2-256	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Verify	Performed on module load.
ECDSA SigGen (FIPS186-5) (A6710)	P-521 with SHA2-512	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Sign	Performed on module load.
ECDSA SigVer (FIPS186-5) (A6710)	P-521 with SHA2-512	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Verify	Performed on module load.
EDDSA SigGen (A6710)	Edwards448, Edwards25519	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Sign	Performed on module load.
EDDSA SigVer (A6710)	Edwards448, Edwards25519	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Verify	Performed on module load.
Hash DRBG (A6710)	SHA2-256	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Instantiate Generate Reseed	Performed on module load.
Counter DRBG (A6710)	AES128bit	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Instantiate Generate Reseed	Performed on module load.
HMAC DRBG (A6710)	SHA2-256	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Instantiate Generate Reseed	Performed on module load.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KAS-ECC-SSC Sp800-56Ar3 (A6710)	P-256	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	Shared secret computation	Performed on module load.
KTS-IFC (A6710)	k=2048	KAT	CAST	Conditional Self-Tests result (1 is success, 0 is error)	SP 800-56B Rev. 2 Encrypt for Basic, SP 800-56B Rev. 2 Decrypt for Basic	Performed on module load.
ECDSA KeyGen (FIPS186-5) (A6710)	-	PCT	PCT	Conditional Self-Tests result (1 is success, 0 is error)	Sign, Verify	Performed on ECC (ECDSA) key pair generation, prior to returning the key pair on conclusion of the call.
EDDSA KeyGen (A6710)	-	PCT	PCT	Conditional Self-Tests result (1 is success, 0 is error)	Sign, Verify	Performed on Edwards (EdDSA) key pair generation, prior to returning the key pair on conclusion of the call.
RSA KeyGen (FIPS186-5) (A6710)	-	PCT	PCT	Conditional Self-Tests result (1 is success, 0 is error)	Sign, Verify	Performed on IFC (RSA, KTS-IFC) key pair generation, prior to returning the key pair on conclusion of the call.

Table 16: Conditional Self-Tests

During the conditional self-tests, data output via the data output interface is prohibited. The cryptographic module does not return control to the calling application until the test is complete. If any of these tests fail, the cryptographic module enters an error state.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A6710)	Message authentication	SW/FW Integrity	On demand	Manually

Table 17: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A6710)	KAT	CAST	On demand	Manually
SHA2-512 (A6710)	KAT	CAST	On demand	Manually
SHA3-256 (A6710)	KAT	CAST	On demand	Manually
AES-GCM-ENC	KAT	CAST	On demand	Manually
AES-GCM-DEC	KAT	CAST	On demand	Manually
AES-CBC (A6710)	KAT	CAST	On demand	Manually
RSA SigGen (FIPS186-5) (A6710)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-5) (A6710)	KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-5) (A6710)	KAT	CAST	On demand	Manually
ECDSA SigVer (FIPS186-5) (A6710)	KAT	CAST	On demand	Manually
EDDSA SigGen (A6710)	KAT	CAST	On demand	Manually
EDDSA SigVer (A6710)	KAT	CAST	On demand	Manually
Hash DRBG (A6710)	KAT	CAST	On demand	Manually
Counter DRBG (A6710)	KAT	CAST	On demand	Manually
HMAC DRBG (A6710)	KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A6710)	KAT	CAST	On demand	Manually
KTS-IFC (A6710)	KAT	CAST	On demand	Manually
ECDSA KeyGen (FIPS186-5) (A6710)	PCT	PCT	-	On Key Pair Generation

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
EDDSA KeyGen (A6710)	PCT	PCT	-	On Key Pair Generation
RSA KeyGen (FIPS186-5) (A6710)	PCT	PCT	-	On Key Pair Generation

Table 18: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error	The self test failure or Pairwise consistency test failure error state	Integrity test failure, KAT failure, or PCT failure	Module load	Returns 0 if integrity test, KAT, or PCT fails.

Table 19: Error States

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

In the cryptographic module, the following two cryptographic boundary modules are built and provided in the shared library of the Crypto API Toolkit, as shown in Figure 1.

- Module of the part corresponding to OpenSSL FIPS Provider
- Module for verifying the operation of cryptographic processing on the part corresponding to OpenSSL FIPS Provider

The build procedure for the cryptographic module is shown below. Note that the build is executed using the test configuration shown in Table 3.

(A) Advance preparation

Use the development environment shown in Table 20.

Item No.	Classification	Product name	Version	Remarks
1	SDK	Intel SGX SDK	2.24	Used in the production of programs for Intel SGX
2	C/C++ compiler	gcc	8.5	Used to build the cryptographic module
3	Go language development environment	Go	1.21.13	Used to build the Crypto API Toolkit

Table 20 Development environment

In this development environment, the following tools should be installed in advance.

(1) build-essential
(2) ocaml
(3) Intel SGX SDK
(4) Intel SGX PSW
(5) libcppunit-dev

The installation method for (1) is shown below. The execution directory is arbitrary.

* Execute as root user.

```
> dnf groupinstall "Development Tools"
> dnf install kernel-devel kernel-headers
```

The installation method for (2) is shown below. The execution directory is arbitrary.

* Execute as root user.

```
> dnf config-manager --set-enabled 8-latest-PowerTools
> dnf install ocaml ocaml-ocamlbuild
```

The installation method for (3) is shown below. The execution directory is arbitrary.

* Execute as root user.

```
> dnf install redhat-rpm-config openssl-devel wget rpm-build git cmake perl python3
> wget https://download.01.org/intel-sgx/sgx-linux/2.24/distro/centos8.3-server/sgx_linux_x64_sdk_2.24.100.3.bin
> bash sgx_linux_x64_sdk_2.24.100.3.bin --prefix=/opt/intel
> vi ~/.profile
   (Added below)
   source /opt/intel/sgxsdk/environment
```

The installation method for (4) is shown below.

* Execute as root user.

```
> cd /opt/intel
> wget https://download.01.org/intel-sgx/sgx-linux/2.24/distro/centos8.3-server/sgx_rpm_local_repo.tgz
> tar xvf sgx_rpm_local_repo.tgz
> yum-config-manager --add-repo file:/opt/intel/sgx_rpm_local_repo
> dnf --nogpgcheck install libsgx-urts libsgx-launch libsgx-epid libsgx-quote-ex libsgx-dcap-ql
```

The installation method for (5) is shown below. The execution directory is arbitrary.

* Execute as root user.

```
> dnf install cppunit-devel
```

(B) Provided files

The provided files used to build the cryptographic module are shown in Table 21.

Item No.	Provided file	Description
1	Upgrade_sgx.patch	Intel SGX SSL, OpenSSL FIPS Provider Change Patch
2	Upgrade_CTK.patch	Crypto API Toolkit Change Patch

3	golang_tool.tar.gz	Go language build tool Generates HMAC values for self-test (integrity test), embedded in the shared library of the Crypto API Toolkit

Table 21 Files provided by the cryptographic module

(C) Build method

Build Intel SGX SSL (including the FIPS Provider of OpenSSL) and Crypto API Toolkit in that order.

(Intel SGX SSL)

* Execute as root user.

(1) Download Intel SGX SSL using the git command.
> git clone https://github.com/intel/intel-sgx-ssl.git -b 3.0_Rev4

(2) Copy "Upgrade_sgx.patch" from item 1 of Table 21 to the intel-sgx-ssl directory.
> cp -p <storage location of the provided files>/Upgrade_sgx.patch intel-sgx-ssl/

(3) Apply the copied "Upgrade_sgx.patch."
> patch -p1 < ./intel-sgx-ssl/Upgrade_sgx.patch

(4) Download openssl-3.2.3 using the wget command.
> wget https://github.com/openssl/openssl/releases/download/openssl-3.2.3/openssl-3.2.3.tar.gz

(5) Move the downloaded "openssl-3.2.3.tar.gz" to the intel-sgx-ssl/openssl_source directory.
> mv openssl-3.2.3.tar.gz intel-sgx-ssl/openssl_source/

(6) Move to the intel-sgx-ssl/Linux directory.
> cd intel-sgx-ssl/Linux

(7) Run make.
> make

(8) Run make install. Library files, header files, and other files necessary for building Crypto API Toolkit are placed in the /opt/intel/sgx directory.
> make install

(9) Move to the directory where (1) was executed.
> cd ../..

(Crypto API Toolkit)

* Execute as root user.

(1) Download the Crypto API Toolkit using the git command.
> git clone https://github.com/intel/crypto-api-toolkit.git

(2) Move to the crypto-api-toolkit directory.
> cd crypto-api-toolkit

(3) Run autogen.sh.
> ./autogen.sh

(4) Run configure.
> ./configure

(5) Move to the directory where (1) was executed.
> cd ..

(6) Copy "Upgrade_CTK.patch" from item 2 of Table 21 to the crypto-api-toolkit directory.
> cp -p <storage location of the provided files>/Upgrade_CTK.patch crypto-api-toolkit/

(7) Apply the copied "Upgrade_CTK.patch."
> patch -p1 < ./crypto-api-toolkit/Upgrade_CTK.patch

(8) Create a new directory called golang_tool and extract "golang_tool.tar.gz" in item 3 of Table 21.
```
> mkdir golang_tool && tar xvzf golang_tool.tar.gz -C golang_tool
```
(9) Copy the files under the golang_tool directory into crypto-api-toolkit/src/p11/trusted.
```
> cp -rp golang_tool/* crypto-api-toolkit/src/p11/trusted
```
(10) Move to the crypto-api-toolkit directory.
```
> cd crypto-api-toolkit
```
(11) Run make.
```
> make
```
(12) Run make install. The libraries required for building and running the application are placed in the /usr/local/lib directory, and the header files are placed in the /usr/local directory.

The following libraries are required to build and run the application.
libp11SgxEnclave.signed.so (internal Enclave shared library)
libp11sgx.so (external Enclave shared library

11.2 Administrator Guidance

(A) Notes on building applications

When building an application that uses the cryptographic module, in addition to the application source files, you must also use the following header files and source files generated by building the Crypto API Toolkit described above.

(Header file)
crypto-api-toolkit/config.h
crypto-api-toolkit/src/p11/untrusted/p11Enclave_u.h

(Source file)
crypto-api-toolkit/src/p11/untrusted/p11Enclave_u.c

(B) Notes on running the application

When executing an application that uses the cryptographic module, you must set the environment LD_LIBRARY_PATH as follows.
```
> export LD_LIBRARY_PATH=/usr/local/lib:/usr/lib64:/opt/intel/sgxsdk/lib64
```

As mentioned in "9.3 SSP Zeroization Methods", the application must zeroize the SSP provided by the cryptographic module through the API. Specifically, the application must zeroize the SSP using the C standard memset function.

(C) Other

- The cryptographic module performs runtime checks related to the application of security parameters, such as minimum key security strength, valid key sizes, and the use of approved curves during public key operations such as signature generation.

- When using AES-GCM, the IVs prepared by the application must meet the following conditions.
  (a) It must be generated using DRBG of the cryptographic module.

(b) The size must be 96 bits or higher.

- When using AES-XTS, the keys prepared by the application must meet the following conditions.
  (a) Key_1 and Key_2 must be separately generated according to the rules for component symmetric keys in SP 800-133 Rev.2, Section 6.3.

11.3 Non-Administrator Guidance

N/A for this module.

11.6 End of Life

To dispose of the cryptographic module, simply delete the shared library of the Crypto API Toolkit.

Note that the storage location of the SSP generated during the use of the cryptographic module is volatile memory, meaning SSPs in this location are not stored persistently.

12 Mitigation of Other Attacks

12.1 Attack List

The cryptographic module implements the following two mitigation techniques against timing-based side-channel attacks.

- Constant-time implementation
- Blinding of values

Constant-time implementation protects cryptographic implementations inside the module from timing analysis. Timing-based side-channel attacks exploit differences in execution time depending on cryptographic operations, but with constant-time implementation, the variation in execution time cannot be traced back to keys, CSPs, or secret data.

The blinding of values protects RSA and ECDSA algorithms from timing attacks. These algorithms are vulnerable to such attacks because attackers can measure the time taken for signature processing or RSA decryption. To mitigate this, the cryptographic module generates a random blinding factor internally and applies it to the input of signature processing or RSA decryption. The blinding factor is discarded after the processing is completed. This makes it difficult for attackers to perform timing attacks on these operations without knowing the blinding factor, preventing them from correlating the time taken for signature processing or RSA decryption with the RSA and ECDSA secret keys.

Appendix1 Glossary

Term	Meaning
AEAD	An acronym for Authenticated Encryption with Associated Data. It is an encryption mode that provides data confidentiality, integrity, and authenticity simultaneously. Also called encryption with authentication or authenticated encryption.

Term	Meaning
AES	An acronym for Advanced Encryption Standard, one of the symmetric key ciphers.
CBC	An acronym for Cipher Block Chaining, one of the AEAD modes. Each plaintext block is XORed with the previous ciphertext block before encryption.
CCM	An acronym for Counter with Cipher Block Chaining-Message Authentication Code, a mode of operation for cryptographic block ciphers. It is a derivation of CTR with authentication using CBC-MAC.
CFB	An acronym for Cipher Feedback, a mode of operation for cryptographic block ciphers. It is characterized by the execution of exclusive OR (XOR) between the ciphertext of the previous block, which is further encrypted, and the plaintext of the current block.
CMAC	An acronym for Cipher-based Message Authentication Code, a cryptographic technique for message authentication. The authentication data is calculated from a symmetric key (MAC key) and the message, based on the block cipher used in symmetric key cryptography.
CRYPTREC	An acronym for Cryptography Research and Evaluation Committee. A project that evaluates and monitors the security of government-recommended cryptography and investigates proper implementation and operation of cryptographic technologies.
CTR	An acronym for Counter, a mode of operation for cryptographic block ciphers. It generates a counter, encrypts it, and XORs it with the plaintext.
ECDH	An acronym for Elliptic Curve Diffie–Hellman key exchange. This is a type of elliptic curve cryptography and public-key cryptography.
FIPS	An acronym for Federal Information Processing Standards. It is a set of information processing standards issued by the U.S. National Institute of Standards and Technology (NIST). FIPS 140-3 is the third version of the document summarizing the security requirements for cryptographic modules.
GCM	An acronym for Galois Counter Mode, one of the AEAD modes. It is a derivation of CTR that generates an authentication code using operations on a Galois field GF(2n).
GMAC	An acronym for Galois Message Authentication Code, a cryptographic technique for message authentication. The authentication data is generated using operations on a Galois field GF(2n).
GPL	An acronym for GNU General Public License. A representative open-source license used for the development and distribution of open-source software.
HMAC	An acronym for Hash Based Message Authentication Code, a cryptographic technique for message authentication. The authentication data is calculated based on a hash function using a symmetric key (MAC key) and the message (data).
NIST	National Institute of Standards and Technology of the United States.

Term	Meaning
OFB	An acronym for Output Feedback, a mode of operation for cryptographic block ciphers. The encrypted initialization vector is XORed with the current plaintext block.
OpenSSL	Open-source software developed and provided for the SSL and TLS protocols.
RSA	One of the classic public-key cryptographic algorithms.
SHA-2	A set of hash function algorithms with six variations: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, and SHA2-512/256.
SHA-3	A set of hash function algorithms with six variations: SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, and SHAKE256.
Mode of operation for cryptographic block ciphers	A mechanism for encrypting messages longer than the block length using block ciphers.
Symmetric key cypher	Encryption where the same key is used for both encryption and decryption. Also called symmetric key encryption.
Block cipher	A type of symmetric key cipher that processes fixed-length data (referred to as blocks) as units.
Public key encryption	Encryption where different keys are used for encryption and decryption. Also called asymmetric key encryption.
Digital signature	A cryptographic technique that enables data integrity and non-repudiation of the signer by generating a signature with a signing key and verifying the validity of the signature with a verification key.
SSP	An acronym for Sensitive Security Parameters, a general term for Critical Security Parameters (CSP) and Public Security Parameters (PSP).
CSP	An acronym for Critical Security Parameters, security-related information whose disclosure or alteration could compromise the security of the cryptographic module.
PSP	An acronym for Public Security Parameters, security-related public information whose alteration could compromise the security of the cryptographic module.
Intel SGX	An acronym for Intel Software Guard Extensions, one of the security functions developed by Intel. It protects data from various unauthorized attacks by encrypting memory with hardware.
Enclave	It refers to a cryptographically secured region in memory, generated by Intel SGX.
Trusted Execution Environment	Also referred to by the acronym TEE. It is a trusted area that is considered safe for sensitive data.
Rich Execution Environment	Also referred to by the acronym REE. It is an untrusted area outside the TEE (Trusted Execution Environment), which is considered unsafe for sensitive data.

Table 22 Glossary

Appendix2 References

Standard	Title & Reference URL
FIPS 140-3	FIPS PUB 140-3 - Security Requirements for Cryptographic Modules March 2019 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf
FIPS 180-4	Secure Hash Standard (SHS) March 2012 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS 180-5	Digital Signature Standard (DSS) February 2023 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
FIPS 202	SHA-3 Standard: Permutation-Based Hash and Extendable-Output Function August 2015 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
FIPS 197	Advanced Encryption Standard November 2001 https://csrc.nist.gov/files/pubs/fips/197/final/docs/fips-197.pdf
FIPS 198-1	The Keyed Hash Message Authentication Code (HMAC) July 2008 https://csrc.nist.gov/files/pubs/fips/198-1/final/docs/fips-198-1_final.pdf
PKCS#1	Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 February 2003 https://www.ietf.org/rfc/rfc3447.txt
SP 800-38A	Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques December 2001 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf
SP 800-38B	NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication May 2005 https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-38b.pdf
SP 800-38C	NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality May 2004 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
SP 800-38D	NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC November 2007 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

Standard	Title & Reference URL
SP 800-38E	Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices January 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf
SP 800-56Ar3	NIST Special Publication 800-56A Revision 2 - Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography May 2013 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf
SP 800-56Br2	NIST Special Publication 800-56B Revision 2 - Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br2.pdf
SP 800-90Ar1	NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators June 2015 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP 800-90B	Recommendation for the Entropy Sources Used for Random Bit Generation January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
SP 800-140B	NIST Special Publication 800-140B - CMVP Security Policy Requirements March 2020] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf

Table 23 References